On the Congestion Responsiveness of Aggregate Internet Traffic: Open-loop vs Closed-loop Session Arrivals

Paper-ID: 303; Number of Pages: 14

Abstract—*A traffic aggregate is congestion responsive if it reacts to network congestion by reducing its rate. The congestion responsiveness of Internet traffic has been largely attributed to TCP's congestion control. In this paper, we argue that congestion control for individual transfers is* not *sufficient to produce responsive aggregate traffic. The offered load at a network link is generated from users/applications that generate finite-length flows or groups of flows (sessions). We examine two session generation models. First, a* closed-loop model *where each user from a certain population can generate a new session only after the completion of her previous session. Second, an* open-loop model *where sessions arrive independently of previous sessions. These two models produce traffic with very different congestion responsiveness, even if each flow is controlled by TCP. We introduce two metrics to quantify the congestion responsiveness of a traffic aggregate, the* throughput responsiveness *and the* flow rate responsiveness*, and show that the closed-loop model results in congestion responsive traffic, while the open-loop model can lead to persistent overload and congestion collapse. We then measure the congestion responsiveness of the traffic at a university access link. These experiments show that both responsiveness metrics are close to zero, which explains why that link is often under persistent overload. We also present an estimation methodology to classify the traffic at a link as open-loop or closed-loop. Our measurements at a dozen of access and core links show that more than 70% of the traffic we analyzed follows the closed-loop model. This implies that a major reason for the congestion responsiveness of Internet traffic may be that most traffic reacts to congestion at the session generation layer.*

I. Introduction

In the past 20 years, the Internet has shown a remarkable ability to handle ever-increasing traffic loads, without relying on resource reservation, per-flow rate allocation, or admission control. As a queueing system, the Internet manages to remain *stable* as long as its traffic is *congestion responsive*. By congestion responsive we mean that the traffic reduces its offered load when it experiences congestion. If the network did not have this self-stabilizing ability, it would be susceptible to *persistent overload*, resulting in a very low per-session goodput and a significant fraction of aborted sessions and applications.

The conventional wisdom is that the TCP protocol, with its congestion control features, is the reason behind the congestion responsiveness of Internet traffic. TCP carries more than 90% of Internet traffic, and each TCP connection reduces its send-window, and thus its throughput, upon an indication of congestion (packet loss). This TCP behavior provides a form of negative feedback from the network to the sources, which is a key element in stabilizing the underlying queueing system. It is believed that if individual transfers react to congestion the way TCP does, then the traffic aggregate at a link will also be congestion responsive. Consequently, there have been proposals to apply similar congestion control algorithms in non-TCP protocols (e.g., TCP-Friendly control [8] or TCP tunnels [18]).

Networking research has primarily focused on *persistent TCP connections*, i.e., transfers that have infinite data to send and that last indefinitely. In this paper, we argue that this modeling assumption can hide some key properties of the congestion responsiveness of Internet traffic. In practice, TCP transfers have a finite size and they are often classified as "elephants" (large file transfers) or "mice" (short, Web-like, transfers). The former carry most of the traffic in the Internet, and for this reason they have been the subject of most previous work in congestion control, *assuming* they can be modeled as persistent transfers [7], [15], [24]. The latter, on the other hand, carry a small fraction of the aggregate traffic, and even though they do not react to congestion the same way elephants do [17], [16], [5], [10], they are often ignored. In this work, we argue that the classification of TCP flows based on their size is not really relevant to the congestion responsiveness of Internet traffic. When we view all TCP flows as *non-persistent* (i.e., with a finite size and duration), then the key issue is *the random process that determines the arrival of flows, rather than packets, into the network*.

Most of the Internet traffic is generated from users or applications that pull (or sometimes push) data from servers or other peers. Each such *session* can generate several TCP connections, and it represents the basic unit of offered load at the session layer of the OSI stack. We identify two fundamentally different session generation models. In the *closed-loop model* (also known as the "interactive model") we have a finite population of users, and each user can generate a new session only after the completion of her previous session. In the *open-loop model* sessions arrive independently of the completion of previous sessions. This model is more appropriate in cases where the population of users is very large and users either do not return to the network, or they do so long after the completion of their last session.

The difference between these two session generation models, in terms of congestion responsiveness, is major. Since closed-loop model users generate new sessions only after the completion of their previous sessions, network congestion will delay the completion of ongoing sessions and thus the generation of the next session from each active user. Consequently, the emergence of congestion regulates the arrival rate of new sessions in the network, resulting in a congestion responsive traffic aggregate. On the other hand, in an open-loop traffic model, sessions arrive independently of each other, and independent of congestion. If the session arrival rate is too high, for a given network capacity and session size, the network can experience persistent overload *even if each individual session uses TCP and is congestion responsive*. The persistent overload will result in a very low goodput for each user, despite the fact that the network bottleneck is fully utilized, and/or a significant fraction of aborted sessions due to user impatience.

In Section III, we introduce two metrics for the congestion responsiveness of the aggregate traffic at a network link: *throughput responsiveness* and *flow rate responsiveness*. The two metrics quantify the degree by which the aggregate traffic at a link reduces its offered load and flow arrival rate, respectively, upon a "canonical congestion event" (specified later). The two metrics should be positive for a congestion responsive traffic aggregate. As we show in Sections IV and V, open-loop TCP traffic has negative throughput responsiveness (even worse than UDP constant-rate flows!) and zero flow rate responsiveness, making it the network's "worse enemy". On the other hand, closed-loop TCP traffic has significantly positive throughput responsiveness (higher than that of persistent TCP connections) and also positive flow responsiveness.

In Section VI, we present direct measurements of the two responsiveness metrics at a university access link. The measurements show that, at least for that link, both the throughput responsiveness and the flow rate responsiveness metrics are often close to zero, implying that most of the traffic follows the open-loop model. This may be the reason why that link often shows signs of persistent overload.

The critical question then becomes whether the Internet traffic mostly follows the open-loop or the closed-loop model. We define the *Closed-loop Traffic Ratio* (CTR) as the fraction of traffic that follows the closed-loop model. Higher CTR implies better congestion responsiveness. In Section VII, we describe a procedure to estimate the CTR at an Internet link using packet traces. Unfortunately, our technique is only applicable to TCP traffic from well understood applications, and so our CTR estimates are applicable to 30%-80% of the traffic, depending on the link that we measure. Measurements at a dozen of Internet links show that the CTR is usually higher than 60%. The CTR measurements imply that a main reason behind the congestion responsiveness of Internet traffic may be that users and applications respond to congestion by slowing down the generation of new sessions.

Finally, in Section VIII we discuss some interesting implications of our conclusions in several areas of networking research and practice. These areas include the usefulness of AQM, admission control and TCP-friendly congestion control, the need for new mathematical and simulation models, and the integration of congestion control in the application or session layers.

II. Related work

The term *congestion collapse* was coined in RFC-896 to identify scenarios where a netwok link provides very low goodput to each user, even though that link is fully utilized. Congestion collapse can occur due to large packet header overheads or due to spurious retransmissions of packets that are still in the network [22]. Another instance of congestion collapse is when a link wastes its capacity transferring packets that are dropped later in the path [7], [6]. In this paper, we are interested in congestion collapse due to a very large number of active flows or sessions. We say that a link experiences *persistent overload* when the offered load remains higher than the capacity over significant time scales, certainly longer than the network Round-Trip Time (RTT). Persistent overload can lead to congestion collapse, if users never abort sessions that take too long. Otherwise, if users are impatient, persistent overload leads to a significant fraction of aborted sessions/applications. Either way, with or without congestion collapse, traffic that causes persistent overload results in poor network performance.

Floyd and Fall [7] and Le et al. [16] proposed to control high-bandwidth flows to prevent persistent overload. Similarly, the Network Border Patrol [1] and ERUF [25] are mechanisms to limit transfers that receive higher throughput than the TCP-friendly rate. The latter is the average throughput that a persistent TCP connection would experience at the same path [7]. Zhao et al. [29] proposed a method to estimate the fraction of congestion unresponsive traffic at a link. In this paper, we show that a traffic aggregate can be congestion unresponsive even if all the constituent flows are TCP or TCP-friendly flows.

S. Ben Fredj et al. [9] considered the open-loop traffic model. They noted that the only reduction in the offered load upon a congestion event is due to aborted transfers. Such transfers, however, result in wasted throughput and user dissatisfaction. For this reason, the authors proposed admission control as the only efficient way to prevent persistent overload. Veciana et al. [4] also considered the open-loop traffic model and concluded that Internet traffic may be unstable under certain conditions. In this paper, we argue that if most of the Internet traffic follows the closed-loop model, then it is congestion responsive and mechanisms such as admission

control may not be necessary for the stability of the Internet.

Previous work in the area of congestion control with closed-loop traffic models is quite limited, mostly due to the challenging underlying mathematical problems. Heyman et al. [11] used a closed-loop traffic model to analyze the performance of Web-like traffic over TCP. They showed that the session goodput and fraction of time the system has a given number of active sessions are insensitive to the distribution of session sizes and "think times", and they only depend on the mean of these distributions. Berger and Kogan [2], as well as Bonald et al. [3], used a similar closed-loop model to design bandwidth provisioning rules for meeting certain throughput-related QoS objectives. In this paper, we show that closed-loop traffic can also have major implications in the congestion responsiveness of Internet traffic.

Most of the previous works with open-loop or closed-loop traffic models assume that TCP congestion control can share the capacity of a link as an ideal Processing Sharing (PS) server [26]. Kherani and Kumar [13] showed that the PS model is not always accurate, mostly because TCP transfers do not manage to keep the link fully utilized under certain conditions. In this paper, we use the PS model just to gain some insight in the congestion responsiveness of the open-loop and closed-loop models. Our simulations, on the other hand, use actual TCP transfers.

Over the last few years, and especially after the seminal work by Kelly et al. [12], several researchers applied control theory to examine the stability of the Internet [15], [28], [19]. A key point about that line of research is that it assumes persistent TCP connections, and so it focuses on the asymptotic stability of the queue size at the network bottleneck. The persistent transfers assumption removes from the problem the importance of the session generation process, which as we argue in this paper, is crucial in determining the congestion responsiveness and stability of Internet traffic.

III. Congestion responsiveness

Traditionally, congestion control has been viewed as a function of the network or transport layer at the OSI stack. Following the end-to-end principle, TCP/IP has adopted congestion control as a transport layer task, implemented at the end-hosts and enforced separately for each TCP connection. The *TCP feedback loop* regulates the offered load (send-window) of a connection, based on the presence of congestion in the network (see Figure 1). The previous view, however, ignores the fact that TCP connections are the result of user and application actions. For example, the TCP connections generated from downloading a Web page, which constitute a "Web session", are the result of a user entering a URL at a web browser or clicking on a link. That user can keep generating new sessions, independent of whether the network is congested or not. In other words, even though the transport layer provides congestion responsiveness through TCP, the session layer can be completely unresponsive if it keeps generating new sessions even when the network is congested. The lack of a *session layer feedback loop* can lead to a large number of active sessions, resulting in very low session goodput and/or aborted sessions.

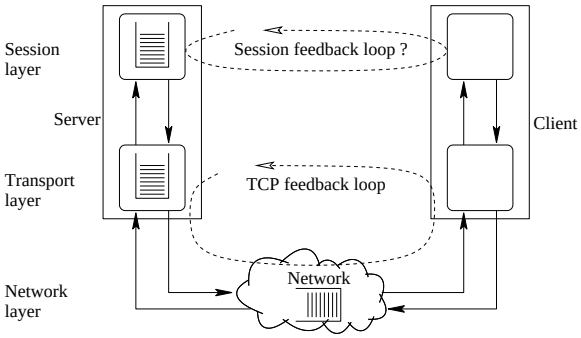

Fig. 1. The TCP feedback loop at the transport layer cannot avoid persistent overload if there is no session layer congestion control.

We do not claim that TCP congestion control is not necessary. It is not sufficient, however, to avoid persistent overload. To understand this point, consider the previous example of a completely unresponsive session layer. When the network becomes congested, each active TCP connection backs-off either reducing its send-window by a large factor or getting into a relatively long silence period (retransmission timeout). This means that congestion control pushes the offered load from each connection back to the TCP buffer of the sender. That connection is still active, however, and so it will keep trying to retransmit any lost packets and to increase its window. As the session layer keeps generating new transfers, the number of competing flows will increase, leading to a diminishing per-session goodput. TCP cannot avoid the emerging persistent overload. Instead, we need a way to tell the session (or application) layer to slow down or stop generating new flows for a while.

To understand the impact of the session layer on the congestion responsiveness of Internet traffic, we focus on properties of the aggregate traffic $\mathcal{G}$ at a bottleneck link $\mathcal{L}$, rather than on properties of the individual flows that constitute $\mathcal{G}$. *Congestion responsiveness* is a general property of $\mathcal{G}$. It relates to whether the traffic in $\mathcal{G}$ reduces its rate when $\mathcal{L}$ is congested. To quantify this property, we introduce two metrics: the *throughput responsiveness* α and the *flow rate responsiveness* β. Both metrics are defined as relative changes in $\mathcal{G}$ after the introduction of a certain *canonical congestion event*, or *stimulus*, at $\mathcal{L}$. A precise specification of the stimulus is important, especially when we compare α or β across different links or time periods.

The throughput responsiveness α of a traffic aggregate $\mathcal{G}$ is defined as the relative reduction in the average input rate of $\mathcal{G}$ upon the introduction of an α-stimulus at $\mathcal{L}$. The α-stimulus that we consider in this paper is a persistent TCP

transfer of a certain RTT that is bottlenecked at $\mathcal{L}$. Let U and U' be the average input rate of $\mathcal{G}$ prior and during the α-stimulus, respectively. The throughput responsiveness α of $\mathcal{G}$ is then defined as

$$\alpha = \frac{U - U'}{U} \tag{1}$$

If $\mathcal{G}$ does not back off after the stimulus, then α is zero. In that case, the stimulus will be able to receive at most the available bandwidth (if any) at $\mathcal{L}$. On the other extreme, α can be as high as 100%, meaning that $\mathcal{G}$ completely shuts down during the stimulus, allowing the latter to capture the entire link capacity. The throughput responsiveness can be also negative, when $\mathcal{G}$ increases its throughput upon the appearance of congestion.

The flow rate responsiveness β of a traffic aggregate $\mathcal{G}$ is defined as the relative reduction in the average flow arrival rate of $\mathcal{G}$ upon the introduction of a β-stimulus at $\mathcal{L}$. The β-stimulus that we consider in this paper is a periodic UDP packet stream of a certain rate and packet size that creates a congestion event at $\mathcal{L}$. Let λ and λ' be the average flow arrival rate in $\mathcal{G}$ prior and during the β-stimulus, respectively. The flow rate responsiveness β of $\mathcal{G}$ is then defined as

$$\beta = \frac{\lambda - \lambda'}{\lambda} \tag{2}$$

When β is zero (or even negative), the flow arrival rate does *not* decrease upon the face of congestion, and so the number of active flows keeps increasing during the stimulus. $\mathcal{L}$ is then susceptible to persistent overload. If β is positive, the flow arrival rate reacts to the congestion event, tending to reduce the offered load at $\mathcal{L}$.

In practice, the measurement of α and β would face the problem of random variations in the input rate and flow arrival rate of $\mathcal{G}$. As shown in Section VI, we should then examine whether the stimulus causes a *statistically significant* variation in the previous two characteristics of $\mathcal{G}$.

IV. Open and closed loop models

The congestion responsiveness of the aggregate traffic $\mathcal{G}$ at link $\mathcal{L}$ depends not only on the congestion responsiveness of the individual flows that constitute $\mathcal{G}$, but also on the session generation process that creates these flows. In this section, we identify and review two fundamentally different session generation models: the *closed-loop* and the *open-loop* model.

To motivate the closed-loop model, consider the access link of a small enterprise with, say N, users. In the inbound direction, most of the traffic at the link is Web downloads that are generated by the activity of these N users. Each user in the "Active" state downloads a Web page (a session), then spends some time in the "Idle" (or "Thinking") state viewing the page, and then either downloads another Web page (the next session) or leaves the system for a longer time period ("Inactive" state). This link would never carry more than N active sessions at a time. Furthermore, if the link becomes congested, then the download latencies of all active sessions will increase, reducing the rate with which new sessions are generated.

To motivate the open-loop model, consider the access link of a Web server. In the outbound direction, the server sends files to a very large population of users located anywhere in the Internet. Assume that a user does not return to this server, at least for a long time, after completing a Web session. Consequently, the server's active sessions are always with new users. This link can carry an arbitrarily large number of sessions (only limited by the number of Internet users), and so it is susceptible to persistent overload. Furthermore, if the link becomes congested, the arrival rate of new sessions will not be affected, as Internet users are typically unaware of the network state in a given path. If new sessions keep coming in, the server's outbound link will eventually experience persistent overload.

In the following, we present the open-loop and closed-loop models more formally, and review some basic mathematical results for the corresponding queueing systems. These results provide a deeper insight in the differences between the congestion responsiveness of the two traffic models. In both cases, we consider a network link $\mathcal{L}$ with capacity C (bytes per second). We assume that the link follows the ideal Processor Sharing (PS) model, meaning that if $N(t)$ sessions are active at time t then each of them receives an instantaneous throughput of $C/N(t)$. The average session size is S (bytes).

A. Open-loop model

The average offered load in the open-loop model is given by λS, where λ is the average session arrival rate. The normalized offered load is defined as $\rho_o = \lambda S/C$. If ρ_o <1, $\mathcal{L}$ is stable and ρ_o is the average utilization. Otherwise, if ρ_o >1, $\mathcal{L}$ becomes unstable if sessions are never aborted [14]. If users are impatient, aborting their active sessions after a certain time period, the underlying queueing system has a finite buffer capacity and so it cannot be unstable [27]. Nevertheless, even if the system is stable, aborted sessions result in user dissatisfaction and poor performance, and so the operating regime where ρ_o >1 should be always avoided [26].

In the case of Poisson session arrivals, the expected service time for a session of size S at a stable open-loop model is given by [13]

$$T(S) = \frac{S}{C(1-\rho_o)} \qquad \text{for} \qquad \rho_o \leq 1 \tag{3}$$

So, the expected throughput for a job of size S is given by the available bandwidth $C(1-\rho_o)$ at $\mathcal{L}$. In terms of the throughput responsiveness α, we see that when $\rho_o < 1$ the expected

throughput of a new transfer (the α-stimulus) is equal to the available bandwidth $C(1-\rho_o)$, and so α=0. Also, the session arrival rate of open-loop traffic remains equal to λ, even after the introduction of the β-stimulus, and so the flow rate responsiveness β for the open-loop traffic model is zero as well. This is true even in the presence of aborted sessions. So, in summary, for the open-loop model we have that

$$\alpha = 0, \quad \beta = 0 \tag{4}$$

B. Closed-loop model

In the closed-loop model we have a fixed number of users N. Each user goes through cycles of activity, with sessions of average size S, followed by idle periods of average length T_i. The average session arrival rate in the closed-loop model is given by

$$\lambda_c = \frac{N}{T_t + T_i} \tag{5}$$

where T_t is the average session completion time. The latter is dependent on the load at $\mathcal{L}$. The average number of sessions at $\mathcal{L}$, for large values of N, is given by [2]

$$E[N_a] = \frac{\rho_c}{1-\rho_c} \quad \text{for} \quad \rho_c < 1 \tag{6}$$

$$= N\left(1-\rho_c^{-1}\right) = N - \frac{CT_i}{S} \quad \text{for} \quad \rho_c > 1 \tag{7}$$

where the normalized offered load is now given by $\rho_c = NS/CT_i$. When $\rho_c \ll 1$, users spend most of the time thinking (i.e., $T_t \ll T_i$), and the system behaves as an open-loop model with session arrival rate $\lambda_c = N/T_i$. However, when ρ_c approaches or exceeds 1, the number of active sessions in the server increases, reducing the average per-session throughput and increasing T_t. The increase in T_t reduces the session arrival rate, as given by (5), keeping the offered load close to the capacity, i.e., $\lambda_c S \approx C$. This means that the closed-loop traffic model is always stable and it cannot experience persistent overload.

When there are N_a active sessions in $\mathcal{L}$, the throughput of a new session (the α-stimulus) will be $C/(N_a+1)$. So, the throughput responsiveness of the closed-loop traffic model (based on the PS model rather than TCP sharing) is

$$\alpha = \frac{1}{N_a+1} \geq \frac{1}{N+1} > 0 \tag{8}$$

In terms of the flow rate responsiveness β, first assume that $\rho_c < 1$. Then, $\lambda_c \approx \rho_c C/S$ because the average session arrival rate is equal to the average session service rate, and the latter is approximately $\rho_c C/S$. Suppose that the β-stimulus causes a reduction of the server capacity from C to $(1-f)C$. For the stimulus to cause congestion, we need to have that $1-f < \rho_c$. Then, the average session arrival rate during the stimulus is $(1-f)C/S$, and so

$$\beta = 1 - \frac{1-f}{\rho_c} > 0 \quad (1-f < \rho_c < 1) \tag{9}$$

When $\rho_c > 1$, the average session arrival rate before the stimulus is C/S, and so

$$\beta = f > 0 \quad (\rho_c > 1) \tag{10}$$

Note that in both cases the responsiveness metric β for the closed-loop model is positive.

C. Mixed traffic

To summarize the previous two models, the open-loop model produces completely unresponsive traffic with α=0 and β=0. On the other hand, the closed-loop model has a strictly positive responsiveness, with $\alpha \geq 1/(N+1)$ and $\beta >$0. Of course, the traffic at an Internet link would be a mix of both open-loop and closed-loop traffic. To characterize such an aggregate, we define the *Closed-loop Traffic Ratio* (CTR) as the fraction of traffic that follows the closed-loop model

$$CTR = \frac{\text{Traffic load from closed-loop model}}{\text{Total traffic load}} \tag{11}$$

The CTR can be between 0 and 100%, with a higher CTR meaning that the aggregate is more congestion responsive.

Assume that U_o and U_c are the offered load from open-loop and closed-loop traffic, respectively. After introducing the α-stimulus, these rates change to U_o' and U_c'. The throughput responsiveness of the traffic mix is then given by

$$\alpha = \frac{(U_o + U_c) - (U_o' + U_c')}{U_o + U_c} \tag{12}$$

From the CTR definition we have that

$$CTR = \frac{U_c}{U_o + U_c} \tag{13}$$

Since the open-loop traffic has zero α, we have that $U_o = U_o'$. So,

$$\alpha = \frac{U_c - U_c'}{U_o + U_c} = CTR * \alpha_c \tag{14}$$

where α_c is the throughput responsiveness of the closed-loop traffic. Therefore, the throughput responsiveness of a traffic aggregate is the throughput responsiveness of the closed-loop component, scaled by the CTR of the mix. Similarly, we can show that the flow rate responsiveness of the traffic mix is:

$$\beta = CTR * \beta_c \tag{15}$$

where β_c is the flow rate responsiveness of the closed-loop traffic.

V. TCP AND UDP TRAFFIC MODELS

In this section, we present analytical and simulation results for the α and β responsiveness metrics for some key traffic models. These traffic models include persistent TCP

transfers, persistent constant-rate UDP streams, open-loop TCP flows, and closed-loop TCP flows. Even though persistent transfers represent a rather unrealistic traffic model, we examine their congestion responsiveness because they have been used extensively in previous work. Also, we want to compare their congestion responsiveness with open-loop and closed-loop TCP traffic. Note that β is not well-defined when the flow arrival rate is zero, and so we ignore that metric in the case of persistent TCP and UDP traffic.

A. Persistent TCP transfers

The throughput of a congestion-limited[1] persistent TCP transfer is determined by the RTT and loss rate at the bottleneck link $\mathcal{L}$. The steady-state throughput of a persistent TCP transfer has been previously derived in [20], [23]. Here, we use the model given by [20]. Assume that the traffic aggregate $\mathcal{G}$ consists of N persistent transfers that have saturated $\mathcal{L}$, introducing a loss rate p. The aggregate throughput U of these transfers is given by

$$U = \sum_{i=1}^{N} \frac{M}{T_i} \sqrt{\frac{3}{2bp}} \tag{16}$$

where T_i is the RTT of the i^{th} transfer, respectively [20]. To derive the throughput responsiveness of $\mathcal{G}$, suppose that after we apply the α-stimulus, the RTT becomes T' and the loss rate becomes p'. Then, α is given by

$$\alpha = 1 - \frac{\sum_{i=1}^{N} \frac{M}{T_i'} \sqrt{\frac{3}{2bp'}}}{\sum_{i=1}^{N} \frac{M}{T_i} \sqrt{\frac{3}{2bp}}} \tag{17}$$

To gain some insight, let us further assume that all TCP transfers have the same RTT, and that the RTT has not increased significantly after we introduced the stimulus, i.e., $T_i = T_i' = T$ for all i. This would be the case if the N connections had saturated the buffers of $\mathcal{L}$ even prior to the stimulus. In this case, the loss rate prior to and during the stimulus are

$$p = \frac{3}{2b} \left(\frac{NM}{CT} \right)^2 \tag{18}$$

$$p' = \frac{3}{2b} \left(\frac{(N+1)M}{CT} \right)^2 \tag{19}$$

and so

$$\alpha = \frac{1}{N+1} \tag{20}$$

Note that, under the previous assumptions, this is the same throughput responsiveness that we would expect based on the PS model.

[1] A persistent TCP transfer is called "congestion-limited" if its congestion window is always smaller than the receiver's advertized window.

B. Persistent constant-rate UDP transfers

We now examine the congestion responsiveness of constant-rate UDP transfers, such as most of the VoIP/Video streams today. Since these transfers are typically unreliable, they do not retransmit any dropped packets. At the same time, they would not reduce their throughput upon the presence of the α-stimulus, and so their throughput responsiveness is α=0.

C. Open-loop TCP transfers

We next consider a traffic aggregate that is generated by open-loop TCP transfers. In Section IV, we showed based on simple results from the open-loop PS queueing model (i.e., ignoring TCP) that the responsiveness metrics α and β are zero. Open-loop TCP traffic, however, has an important difference with the corresponding PS model. Specifically, TCP retransmits every dropped packet; recall that the PS model is fluid-based and it does not drop packets. This means that the average offered load from an open-loop TCP traffic aggregate would *increase* during the α-stimulus, as long as the latter causes further congestion and increased loss rate. Furthermore, TCP can suffer from redundant retransmissions, meaning that it resends packets that were not dropped. Retransmissions, necessary or redundant, make *the throughput responsiveness of open-loop TCP traffic negative*. This is interesting, as it implies that open-loop TCP traffic is even less congestion responsive than constant-rate UDP traffic!

D. Closed-loop TCP transfers

The expected throughput of a TCP flow in the closed-loop model is given by

$$R_c = \frac{S}{D_S(p,T) + T_i} \tag{21}$$

where S is the average transfer size, T_i is the average idle time between successive transfers, and $D_S(p,T)$ is the expected duration of a transfer of size S given an RTT T and a loss rate p. If the user population size is N, then the expected throughput of the aggregate is $U = NR_c$. Suppose that, after the α-stimulus, the RTT increases to T' and the loss rate increases to p'. Then, the throughput responsiveness of the traffic aggregate is given by

$$\begin{aligned} \alpha &= \frac{N(R_c - R_c')}{NR_c} \\ &= \frac{D_S(p',T') - D_S(p,T)}{D_S(p',T') + T_i} \end{aligned} \tag{22}$$

We performed a number of ns-2 simulations with closed-loop TCP traffic. The simulations use a simple dumbbell topology where both the traffic aggregate $\mathcal{G}$ and the stimulus are connected to the bottleneck link $\mathcal{L}$ via high capacity and low delay links with large buffers. The capacity C of

$\mathcal{L}$ is 50Mbps and its buffer size B is set to 1.2 times the bandwidth-delay product of the path. The minimum RTT is 100msec, while the maximum queueing delay at $\mathcal{L}$ is 120msec. The packet size is 1500 bytes. The TCP flows use the Reno ns-2 implementation with Selective Acknowledgments, and they all have the same RTT. The flow start times are uniformly distributed in the simulation interval. The average flow size is 18 packets, and the flow size distribution is uniform between 16 and 20 packets. The idle time is uniformly distributed between 1 to 3 seconds (T_i=2sec). We control the offered load in this model by adjusting the total number of simulated users N.

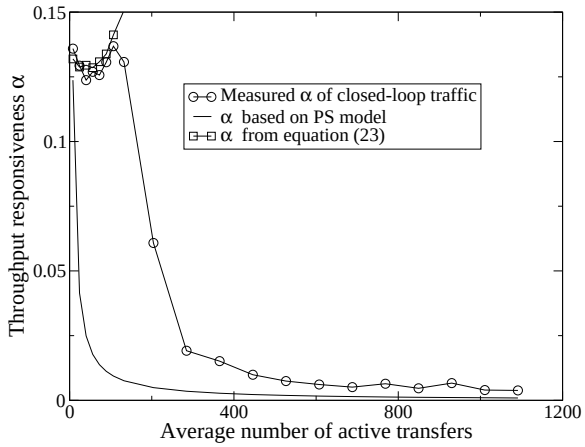

Fig. 2. Throughput responsiveness of closed-loop TCP transfers.

Figure 2 shows the throughput responsiveness α as a function of the average number of active flows N_a.[2] When N_a is less than about 200 flows, $\mathcal{L}$ is not congested prior to the stimulus (p=0) and the loss rate after the stimulus is also very close to zero $p' \approx 0$. In that case, the duration of each TCP flow is determined by its RTT. Suppose that a transfer with size S takes k_S RTTs to complete without any packet drops. Then, if the RTT increases from T to T' after the stimulus, the duration of each transfer will increase from $D_S(p,T) = k_S T$ to $D_S(p',T') = k_S T'$. So,

$$\alpha = \frac{k_S(T' - T)}{k_S T' + T_i} \tag{23}$$

This formula accurately predicts the throughput responsiveness in Figure 2 (around 0.12-0.14, depending on the variations of T' with N_a) as long as N_a is less than 200 flows.

When the average number of active flows is more than 200, $\mathcal{L}$ is congested even before the stimulus. In that case, the introduction of the α-stimulus means that a new (persistent) TCP flow starts competing for bandwidth in a congested link with N_a other (short) TCP flows. Both the stimulus and the short TCP flows have the same RTT. This scenario may seem similar at first with $N_a + 1$ competing persistent TCP connections, in which we would expect that α=$1/(N_a + 1)$ (see Equation 20). As Figure 2 shows, however, this is not the case. The congestion responsiveness of closed-loop TCP flows is higher than that of persistent TCP connections. This is because the short closed-loop TCP flows cannot recover from packet losses as quickly as the longer TCP stimulus. Specifically, short TCP flows experience more retransmission timeouts and they cannot use Fast Retransmit as often as long TCP flows do because they have a smaller window [21]. In summary, *closed-loop TCP traffic is more responsive, in the presence of congestion, than persistent TCP flows*.

[2] A larger N corresponds to a larger N_a. We directly control N, while N_a is measured at the end of the simulation.

VI. Responsiveness measurements

In this section, we present direct measurements of α and β collected at the Internet access link of a university ABC[3]. This access link has a 30Mbps capacity, and is congested for several hours each day. We are able to collect bidirectional packet traces at that link using *tcpdump*. Furthermore, we can cause α-stimuli, performing long congestion-limited TCP transfers between a host within that university and a host at another university XYZ[3], and β-stimuli, performing periodic UDP transfers of a certain rate and packet size at the same path. Our stimuli can cause congestion at the access link, and so we can directly measure α and β based on the definitions (1) and (2), respectively. In the following, we describe the measurement methodology and statistical analysis that we followed to estimate the two congestion responsiveness metrics in a number of experiments.

A. Measurement methodology

We generate a sequence of α-stimuli and β-stimuli to estimate a statistically significant α and β, respectively. Each stimulus transfer lasts for 3 minutes, followed by 3 minutes of inactivity before we generate the next stimulus. Simultaneously we collected packet traces at the access link. From these traces, we measured the cross-traffic throughput or the flow arrival rate in one-minute intervals for the throughput or the flow rate responsiveness, respectively. The rates for the third minute, without and with stimulus, are taken as the rates before and during stimulus, respectively, for responsiveness calculations. A total of 30 transfers were performed, providing us 30 samples of α (or β) in each experiment.

To examine whether the measured responsiveness is positive with some confidence, we performed the following statistical analysis. Using the non-parametric sign (or Fisher) test, we examine the null hypothesis that α=0 (or β=0) by comparing the throughput (flow arrival rate) measurements before and during the stimulus. The alternative hypothesis is that the responsiveness metric is positive. When the p-value

[3] To preserve anonymity, we do not disclose the name of the university.

is low, typically below 5%, we can reject the null hypothesis with high confidence. When that is the case, we also report a 95% confidence interval for α (or β). A positive confidence interval of these responsiveness metrics is an indication of closed-loop traffic. On the other hand, when the p-value is not so low, the responsiveness does not appear significantly positive, which is an indication of open-loop traffic.

In the following, we show the measurement results for throughput and flow rate responsiveness at ABC's access link.

B. Results

The passive monitor for this university is located between the university and the tight link. Therefore, we observe cross-traffic's input rate for the outbound traffic. In the inbound direction, however, what we observe is the cross-traffic rates at the *output* of the tight link, which will not be the same as the cross-traffic's input rate during congestion. Since throughput responsiveness is defined in terms of the cross-traffic's *input rate*, in the following we measure α for outbound direction only. On the other hand, flow arrival rate remains the same before and after the tight link, therefore we can measure β for both the directions.

B.1 Throughput responsiveness

In order to measure throughput responsiveness, we introduced $\alpha-$stimulus, a congestion-limited TCP transfer and estimate α as outlined above. Table I shows the $p-$value of

Experiment	Direction	p-value	95% conf-interval
α-1	Outbound	0.819	-
α-2	Outbound	1.000	-
α-3	Outbound	0.572	-
α-4	Outbound	0.900	-
α-5	Outbound	0.428	-
α-6	Outbound	0.428	-
α-7	Outbound	1.000	-
α-8	Outbound	0.708	-
α-9	Outbound	0.021	0.000 - 0.044
α-10	Outbound	0.100	-

TABLE I
THROUGHPUT RESPONSIVENESS α.

the sign test and the 95% confidence interval for the cases where $p-$value is smaller than 0.05. We note that in most cases the $p-$value is much higher than 0.05, implying that α is not significantly positive. Furthermore, in the only case where $p-$value is small, the confidence interval for α is very closed to zero (run α-9). Figure 3 shows this experiment where the α-stimulus appears to have a significant effect on the aggregate throughput of the traffic at the give link. We observe that cross-traffic rate reduces by a small amount in many cases, providing a positive, however small, α estimates.

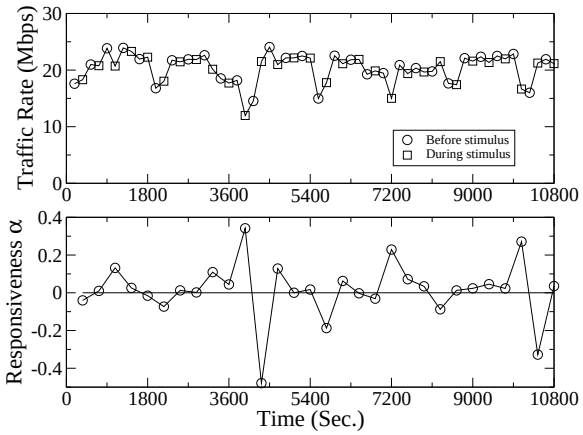

Fig. 3. Experiment α-9.

The given link appears to have zero or very low throughput responsiveness most of the time, implying that most of its traffic follows the open-loop model. This may be the reason why this link remains under persistent overload for several hours each day.

B.2 Flow rate responsiveness

We followed a similar methodology to measure the flow rate responsiveness β. In this case, a β-stimulus is a UDP periodic stream that captures approximately 20% of the link's capacity with 1470-byte packets. We performed 10 experiments for each direction of the link.

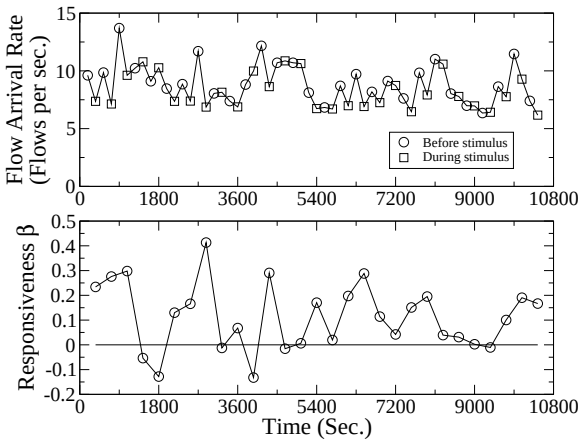

Fig. 4. Experiment β-12.

Figure 4 shows an experiment where the β-stimulus appears to have a significant effect on the flow arrival rate of the traffic at the give link. Table II shows the resulting p-values for all the experiments. Only three of the twenty experiments showed a statistically significant positive flow rate

Experiment	Direction	p-value	95% conf-interval
β-1	Inbound	0.356	-
β-2	Inbound	0.356	-
β-3	Inbound	0.575	-
β-4	Inbound	0.004	0.045 - 0.156
β-5	Inbound	0.645	-
β-6	Inbound	0.068	-
β-7	Inbound	0.181	-
β-8	Inbound	0.008	0.026 - 0.126
β-9	Inbound	0.292	-
β-10	Inbound	0.292	-
β-11	Outbound	0.049	-
β-12	Outbound	0.001	0.031 - 0.170
β-13	Outbound	0.708	-
β-14	Outbound	0.292	-
β-15	Outbound	0.292	-
β-16	Outbound	0.819	-
β-17	Outbound	0.708	-
β-18	Outbound	0.100	-
β-19	Outbound	0.428	-
β-20	Outbound	0.572	-

TABLE II
FLOW RATE RESPONSIVENESS β.

responsiveness β. As in the case of α, this implies that most of the traffic at the given link follows the open-loop model.

VII. CTR MEASUREMENTS

The previous section presented direct measurements of congestion responsiveness from a congested university access link. Unfortunately, we do not have access to packet traces from other links that are either congested, or that would become congested after we apply an α or β stimulus. Note that without congestion during the stimulus, we do not have a way to measure α or β. Consequently, we decided to measure the congestion responsiveness in other Internet links *indirectly*, estimating the CTR of their traffic. As discussed in Section IV, the CTR is an indirect indicator of congestion responsiveness. In this section, we analyze packet traces from several Internet links attempting to classify their traffic as open-loop or closed-loop, and thus to estimate the CTR.

The outline of the CTR estimation methodology is as follows. First, we need to partition the packet trace into a set of sessions initiated by each user. This process is far from simple, and it requires some knowledge of the corresponding application. For example, in the case of HTTP download, a "user" is associated with a specific IP *destination* address. Each session corresponds to a download operation that that user requested and can consist of multiple "transfers". After we have transformed the packet trace into a "session trace", we then classify each session as open-loop or closed-loop.

A newly arriving session a user is considered closed-loop if its arrival is dependent on the progress of the previous session from that user, else it is an open-loop session. Such classification requires knowledge of user actions/thoughts between two sessions and can be accurately performed only by an Oracle. Without a monitor with oracular view, we make an assumption that maps the dependence of session arrivals from a user to the arrival time of the new session with respect to the finish of previous session from that user. Specifically, a session from a user is dependent on her previous session, and should be classified as closed-loop, if it starts soon after the finish of the previous session. If the next session from a user starts while the previous session is still active or after a long time from the finish of the previous session, then the new session is independent of previous session and should be classified as open-loop.

We note that the termination of the previous session is neither necessary nor sufficient condition for a new session from the same user to be dependent on the previous one. For example, a user obtains a URL through one web-page download (session X) and starts downloading its content in another window (session Y) while session X is still active. In this case, the arrival of the session Y is dependent on the progress of the session X, as it is the session X that brings the URL needed to start the session Y. However, session Y starts before X finishes. Similarly, a user can start two sessions independently in two separate windows/tabs. However, the first session has small amount of data to transfer and just happens to finish before the second one starts. Therefore, the assumption above will inaccurately classify the second session in both the examples. We don't expect the error due to these effects to be large since not many independent sessions from a user with multiple window/tab will arrive after previous session from that user finished (latter case). Similarly, not many dependent sessions will be opened in new window/tab such that the start of the next session does not terminate the existing session (former case). Thus our approach will provide us a CTR estimate that is representative of the characteristic of the traffic aggregate.

To summarize, the first session from a user is always classified as open-loop. A subsequent session is classified as closed-loop, as long as it starts after the completion and within a "maximum think time" from the completion of the last session of that user. Otherwise, we assume that the session arrival is independent of the previous session or the user was inactive for some time and the new session is again classified as open-loop. Finally, after we have classified each session as open-loop or closed-loop, we count the bytes from each type and calculate the CTR.

A. Definitions and methodology

We start with a more detailed explanation of the key terms and of the CTR estimation methodology for the case of the HTTP/HTTPS downloads. HTTP is not the only protocol/application for which we can perform the previous analysis. Traffic with other well-known ports is also well understood, in terms of who is the "user", what constitutes a "session", etc, and so we also applied the previous methodology for those applications. Unfortunately, a large part of Internet traffic today does not use well-known ports. That traffic is probably generated by peer-to-peer applications. Eventually, we were able to analyze more than 50% of the TCP traffic in half of the traces we analyzed. In some traces the fraction of traffic we could analyze was as high as 78%, but in others was as low as 26%.

Users and sessions: *User* is an entity (typically a person, but it can also be a automated process) that issues Web requests. Each such application-layer request is a *session*. We assume that a user is identified in the packet trace by a certain destination IP address. An important exception to this rule is when multiple users share the same host (e.g., remote login) or when the host addresses of different users are translated somewhere in the network to the same IP address (e.g., NATs or proxies). We have devised a heuristic that can identify multi-user hosts, described in the Appendix. Multi-user hosts are ignored in the rest of our analysis.

In HTTP/HTTPS, a user is associated with a destination address, because users typically download traffic. In applications that mostly upload traffic to remote hosts, the user would be associated with a source address. A download session, associated with a certain user, can contact a number of different servers, and it can consist of several TCP connections with different destination ports. Consequently, the traffic that belongs to a certain session would have the same IP destination address, but potentially different source addresses and/or destination ports.

Connections and transfers: A TCP connection is identified in the packet trace by a unique 5-tuple field (Source IP address, Destination IP address, Source Port, Destination Port and Protocol). A connection has certain start and finish times, corresponding to the timestamps of the first and last packets in the connection, respectively. We ignore connections that were ongoing at the start or end of the trace. Pure ACKs are packets without payload. A connection with more pure ACKs than data segments is considered an *ACK flow* and it is ignored from our analysis.

HTTP 1.1 introduced persistent connections, meaning that a connection can stay alive for a long time, transferring Web objects that belong to different sessions. We partition a connection into one or more *transfers*, with different transfers being part of different sessions. Figure 5 shows an example of a persistent connection that includes two transfers. The packet interarrivals within the same transfer are determined by TCP (e.g., self-clocking, retransmission timeouts) or network delays. The packet interarrivals between different transfers, however, are typically determined by the latency of user actions (e.g., clicking at a Web link). Consequently, the inter-transfer interarrivals ("gaps") are usually much longer than the intra-transfer interarrivals. We use this observation to partition a connection into transfers. If a packet interarrival within the same connection is larger than a certain *Silence Threshold* (STH), which represents the maximum intra-transfer gap, then a new transfer starts with that packet.

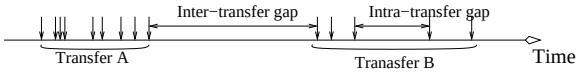

Fig. 5. Packet interarrivals from two transfers within the same connection.

To choose a reasonable value for STH, we examined values in the range 1sec-1min. A very small STH would partition a transfer in smaller chunks, while a very large STH would merge different transfers together. So, we expect that the average transfer size would increase with STH. More importantly, we expect that for a certain range of this threshold, when it falls between the larger intra-transfer gaps and the lower inter-transfer gaps, the number of transfers would be almost constant. Figure 6 shows the median transfer size as a function of STH for the XYZ-in packet trace. Note that the median transfer size is roughly constant when the threshold is between 35-45sec. In the following, we set STH to 40sec.

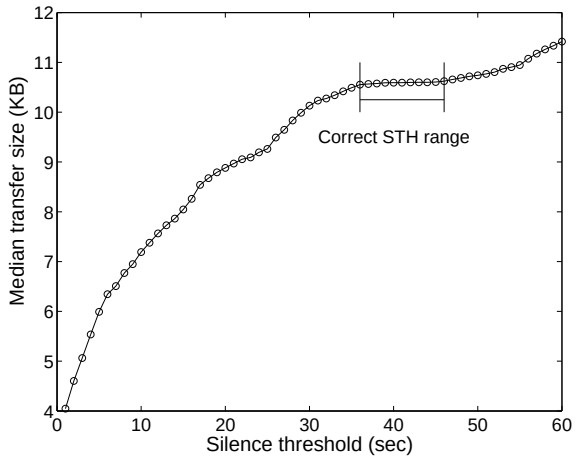

Fig. 6. Median transfer size as a function of the Silence Threshold STH.

Grouping transfers into sessions: Since a session can consist of several transfers, as shown in Figure 7, we need to identify the transfers (TCP connections or segments of TCP connections) that were generated as a result of the same user action. The key observation here is that the interarrival of two transfers that belong to the same session will typically

be much lower than the interarrival of transfers that belong to two different sessions. The latter are separated by the latency of a user action. We expect transfers of different sessions to start at least one second or so from each other, while transfers that belong to the same session are typically generated automatically by the Web browser within tens or hundreds of milliseconds.

Specifically, if the interarrival between two successive transfers is larger than a certain parameter, referred to as *Minimum Session Interarrival* (MSI), then the second transfer starts a new session. We examine the robustness of the CTR estimate to the exact choice of the MSI in Section VII-B.

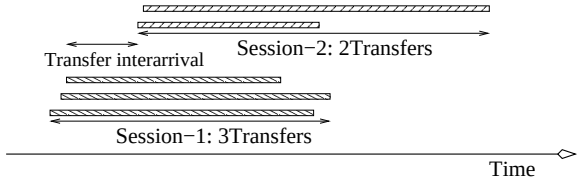

Fig. 7. Timeline of two sessions. Each session consists of several transfers.

Classifying sessions as open-loop or closed-loop: After we have transformed the packet trace into a "session trace", we now classify each session as open-loop or closed-loop. Recall that the key difference between these two is that in the open-loop model sessions arrive independent of the progress of previous sessions from the same user.

The first session from a user is considered open-loop, given that that user has no dependency to any previous sessions. If that user generates a new session after her previous session finishes and no later than the *Maximum Think Time* (MTT), then the arrival of this new session is considered dependent on the progress of the previous session. So, we classify that session as closed-loop. If, however, the new session arrived during her previous session or much later, more than MTT, we assume that the user either does not wait for her previous session's progress or was inactive for some time and now she returns to the network without any "memory" of her previous sessions. Thus, we classify that session as open-loop. The cases are shown in Figure 8. We examine the robustness of the CTR estimate to the exact choice of the MTT in Section VII-B.

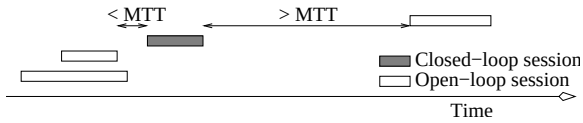

Fig. 8. Classifi cation of different sessions from the same user as open-loop or closed-loop.

Other traffic with well-known ports: For other traffic, not generated by HTTP/HTTPS, we followed the convention that the transfer is an upload if the destination port is well-known, and a download if the source port is well-known. The rest of the estimation methodology is the same as in Web traffic.

CTR calculation: Once we have classified sessions as open-loop or closed-loop, we then calculate the CTR as the fraction of bytes from closed-loop sessions.

B. Robustness of estimation methodology

The CTR estimate depends on the following parameters: STH, MSI, and MTT. We have already shown that a robust value for STH is around 40sec. In this section, we investigate the optimal range for MSI and MTT, and examine the CTR's robustness to the choice of these two parameters.

The MSI is used to merge together different transfers of the same session. As these transfers are typically machine-generated, they start almost simultaneously. In the packet trace, however, they can appear with slightly longer spacings due to network delays. A reasonable range for this parameter is between 0.5-1sec. The MTT, on the other hand, represents the longest "thinking" time for a user during Web browsing, before we consider that user inactive. A reasonable range for this parameter is between 5-30min.

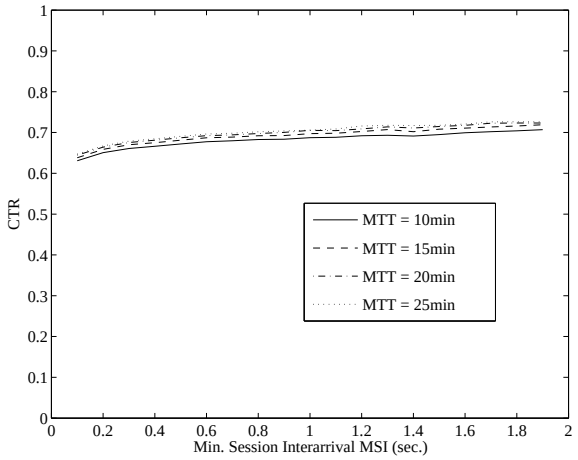

Fig. 9. Variation of CTR for different MSI and MTT values.

Figure 9 shows the CTR for the XYZ-in packet trace for different MSI and MTT values. We see that the CTR does not significantly depend on these parameters (it varies in a small range between 0.6-0.72) as long as the MSI and the MTT fall between 0.5-2sec and 5-25min, respectively.

To further examine the robustness of CTR on the two thresholds, we performed two-factor analysis of variance. The null hypothesis is that the CTR is independent of these two parameters. We can reject this hypothesis at a significance level of 0.05. However, the hypothesis that the CTR is independent of the interaction of these two parameters cannot be rejected at the same significance level.

To quantify the dependence of CTR on the MSI and MTT values we performed linear regression for the CTR with re-

spect to both parameters. We found that the slope of the CTR with respect to MSI is 0.0232/sec, and the slope with respect to MTT is 0.0020/min. Therefore, we concluded that the CTR is insensitive to these parameters in the ranges that we consider. In the rest of the analysis, *we set MSI=1sec and MTT=15min*.

C. Results

We performed a weeklong measurement of CTR at the access link of ABC. The results show a significant variation of CTR at different time of day. In both the directions, CTR varies from as low as 0.25 to as high as 0.82. The majority of measurements for inbound direction lies in the range 0.6 to 0.7. Which in the outbound direction is in the range 0.4 to 0.5. This link, however, has very different traffic composition than other traces we analyzed. The fraction of well known port traffic is much lower than any other trace we analyzed. Upon further analysis of different port used in this trace, peer-2-peer applications seem to dominate bytes transferred.

Table III summarizes the metadata of the traces that we used in this paper. The traces are from university access links, commercial access links and backbone links, and they were collected between 2001-2005. It also shows the CTR estimates for Web and other well-known port traffic in these 12 Internet traces we analyzed.

An important observation is that the CTR for access links is always higher in the inbound direction than in the outbound direction. This can be explained by the fact that users that initiate sessions in the inbound direction belong to the limited population of users within that campus network. On the other hand, users that initiate sessions in the outbound direction come from all over the Internet and they belong to a much larger population. Consequently, the fraction of open-loop traffic in the latter is higher (lower CTR).

The most important observation, however, is that the CTR for almost all traces is high, and more than 60% in all traces but two. Even the backbone links, where we would expect more open-loop traffic due to the large number of users have a high CTR. This implies that a major reason for the congestion responsiveness of Internet traffic may be that most applications follow the closed-loop model, and so they are responsive to congestion at the session generation layer.

VIII. Conclusions and discussion

First, the assumption of persistent TCP connections hides certain key issues about the congestion responsiveness of Internet traffic. Persistent TCP transfers are always responsive and they cannot lead to persistent overload. Second, we identified the session generation process as a crucial factor for the congestion responsiveness of aggregate traffic. If this process receives feedback from the network, at the session layer, then the resulting traffic can be modeled as closed-loop and it is congestion responsive. Otherwise, if the traffic at a certain link follows the open-loop session generation model, that link is at the risk of persistent overload and congestion collapse. Third, our direct measurements of congestion responsiveness at a university access link show that, at least for that link, both the throughput responsiveness and the flow rate responsiveness metrics are often close to zero, implying that most of the traffic follows the open-loop model. This may be the reason why that link often shows signs of persistent overload. Fourth, our traffic measurements, even though limited in terms of both the fraction of traffic we can classify and the number of links we examined, show that more than 60% of the traffic we analyzed follows the closed-loop model. These CTR measurements imply that a main reason behind the stability and congestion responsiveness of the Internet may be that users and applications respond to congestion by slowing down the generation of new sessions.

In the following, we discuss some more implications of this work in specific areas of networking research and practice.

A. AQM and network stability

Active queue management (AQM) mechanisms, such as RED, REM, PI controllers, etc., have been proposed as a way to provide stability in the Internet. It is important to note that such stability studies assume persistent TCP connections. With that traffic model, the AQM mechanisms can control and stabilize the queue length and the bottleneck link utilization. The effectiveness of AQM mechanisms with non-persistent traffic, however, is much less understood. As we showed in this paper, the offered load of open-loop TCP traffic does not depend on network state. AQM mechanisms cannot regulate such an aggregate, and they will be unable to avoid persistent overload if the offered load exceeds the network capacity.

B. Is admission control necessary?

Several researchers advocate the use of admission control as the only way to regulate the offered load and avoid the congestion collapse risk. We agree with them, if the traffic is mostly open-loop. Without admission control, the only way to avoid congestion collapse is to expect that users will be impatient and they will abandon very slow ongoing transfers. This is not an efficient way to control congestion however. Admission control, on the other hand, can limit the number of active sessions or flows in the network and it can provide a throughput guarantee to each of them. We also showed, however, that more than 60% of the Internet traffic that we analyzed is *not* open-loop. This implies that admission control may not be necessary, as long as the CTR at the given link is sufficiently high.

Trace_ID	Direction	Collection time	Link location	Duration	TCP traffic		
					Total	Well-known ports	
					GB (%)	bytes	CTR
XYZ-in	In	07-Jan-05	XYZ	2Hr.	129.74 (97.15)	63.50%	0.71
XYZ-out	Out	07-Jan-05	XYZ	2Hr.	208.06 (98.92)	47.90%	0.57
Los-Nettos	Core	03-Feb-04	Los-Nettos, CA	1Hr.	59.37 (94.96)	65.59%	0.77
UNC_em0	Out	29-Apr-03	UNC	1Hr.	153.19 (97.33)	35.78%	0.61
UNC_em1	In	29-Apr-03	UNC	1Hr.	41.51 (87.76)	26.59%	0.76
UNC_em1_2	In	24-Apr-03	UNC	1Hr.	55.25 (85.67)	44.88%	0.78
MFN_0	Core	14-Aug-02	MFN, San Jose	1Hr.	151.38 (96.31)	61.09%	0.69
MFN_1	Core	14-Aug-02	MFN, San Jose	1Hr.	186.93 (97.75)	71.83%	0.62
IPLS_0	Core	14-Aug-02	Abilene	1Hr.	172.22 (96.40)	41.93%	0.70
IPLS_1	Core	14-Aug-02	Abilene	1Hr.	177.99 (85.00)	47.27%	0.64
Auckland_0	In	06-Nov-01	Auckland, NZ	6Hr.	0.58 (94.83)	72.99%	0.73
Auckland_1	Out	06-Nov-01	Auckland, NZ	6Hr.	1.44 (98.43)	77.99%	0.67

TABLE III

CTR OF THE ANALYZED TCP TRAFFIC AT VARIOUS LINKS.

C. TCP-friendly congestion control

The use of TCP-friendly congestion control has been encouraged in all non-TCP protocols and applications. The basic motivation for such proposals is that TCP-friendly transfers can avoid congestion collapse. We have shown, however, that even if a traffic aggregate consists entirely of TCP connections, it can still cause congestion collapse or persistent overload if it is open-loop. The same is obviously true for TCP-friendly traffic. Therefore, the use of TCP-friendly congestion control is not sufficient to guarantee stability. On the other hand, TCP-friendly congestion control is important and beneficial as a way to improve fairness in the bandwidth sharing among TCP and non-TCP transfers.

D. Traffic engineering and network provisioning

Traffic engineering, as well as other provisioning mechanisms, require estimates for the amount of traffic flowing between any inbound/outbound points in a network. Furthermore, such mechanisms assume that if a given traffic aggregate is switched from one route to another, then the throughput of that aggregate will *not* change. This assumption is not true for TCP persistent connections. It is well understood that the throughput of such transfers depends on the RTT and loss rate in the underlying path, raising concerns for the applicability of traffic engineering.

On the other hand, the offered load from open-loop TCP traffic does not depend on the underlying network path, making such traffic consistent with common assumptions in traffic engineering. The same is true for closed-loop TCP traffic, as long as the offered load remains below the capacity of the underlying paths.

E. New traffic models for simulations and analysis

Most of the previous research in congestion control assumed persistent TCP flows in both simulation and analysis. We believe that the community should abandon that assumption and adopt non-persistent traffic models instead. It is also important that these models consider a mix of both open-loop and closed-loop TCP traffic, with realistic (measurement-based) CTR values. Especially, in the case of closed-loop traffic, the mathematical results are quite limited; more research in that direction would be valuable.

F. Session layer congestion control

At the more practical side, we recommend that all network applications use some form of congestion control at the session layer. This can be as simple as adopting one of the following rules: do not generate a new session until the previous session has completed, slow down the generation of new sessions if the network appears to be congested, or do not keep more than a certain number of sessions active. Some applications already follow similar rules.

It is also important that session layer congestion control is implemented in applications that generate transfers automatically, without user intervention. For example, NNTP servers transfer news to their peers periodically, independent of whether the underlying network is congested or not. CDN servers also perform such periodic transfers. Effectively, such applications generate open-loop TCP traffic, raising the possibility for congestion collapse if their aggregate load is comparable to the underlying capacity.

REFERENCES

[1] C. Albuquerque, B. J. Vickers, and T. Suda. Network Border Patrol: Preventing Congestion Collapse and Promoting Fairness in the Inter-

net. *Transactions on Networking*, 12(1), 2004.
[2] A. Berger and Y. Kogan. Dimensioning Bandwidth for Elastic Traffic in High-Speed Data Networks. *IEEE/ACM Transactions on Networking*, 8(5):643–654, 2000.
[3] T. Bonald, P. Olivier, and J. Roberts. Dimensioning High Speed IP Access Networks. In *18th International Teletraffic Congress*, 2003.
[4] G. de Veciana, T. Lee, and T. Konstantopoulos. Stabily and Performance Analysis of Networks Supporting Services. *IEEE/ACM Trans. on Networking*, 9(1), 2001.
[5] S. Ebrahimi-Taghizadeh, A. Helmy, and S. Gupta. TCP vs. TCP: a Systematic Study of Adverse Impact of Short-lived TCP Flows on Long-lived TCP Flows. In *Proceedings of IEEE INFOCOM*, 2005.
[6] S. Floyd. *Congestion Control Principles*, Sept. 2000. IETF RFC 2914.
[7] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–473, Aug. 1999.
[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of ACM SIGCOMM*, 2000.
[9] S. B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J. W. Roberts. Statistical Bandwidth Sharing: A Study of Congestion at Flow Level. In *Proceedings of ACM SIGCOMM*, Aug. 2001.
[10] L. Guo and I. Matta. The War Between Mice and Elephants. In *Proceedings of IEEE ICNP*, 2001.
[11] D. Heyman, T.V.Lakshman, and A. L. Neidhardt. A New Method for Analysis Feedback-Based Protocols with Applications to Engineering Web Traffic over the Internet. In *ACM SIGMETRICS*, pages 24–38, 1997.
[12] F. P. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 49:237–252, 1998.
[13] A. A. Kherani and A. Kumar. Stochastic Models for Throughput Analysis of Randomly Arriving Elastic Flows in the Internet. In *Proceedings of IEEE INFOCOM*, 2002.
[14] L. Kleinrock. Time-shared Systems: A Theoretical Treatment. *Journal of the ACM*, 14(2):242–261, 1967.
[15] S. Kunniyur and R. Srikant. Stable, Scalable, Fair Congestion Control and AQM Schemes that Achieve High Utilization in the Internet. *IEEE Transactions on Automatic Control*, 49:2024–2029, 2004.
[16] L. Le, J. Aikat, K. Jeffay, and F. Smith. Differential Congestion Notification: Taming the Elephants. In *Proceedings of ICNP*, 2004.
[17] L. Le, J. Aikat, K. Jeffay, and F. D. Smith. The Effects of Active Queue Management on Web Performance. In *Proceedings of SIGCOMM*, 2003.
[18] B. P. Lee, R. K. Balan, L. Jacob, W. K. G. Seah, and A. L. Ananda. Avoiding Congestion Collapse on the Internet using TCP Tunnels. *Computer Networks*, 39(5):207–219, 2002.
[19] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. Doyle. Dynamics of TCP/RED and a Scalable Control. In *INFOCOM*, 2002.
[20] M. Mathis, J. Semke, J. Madhavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *ACM Computer Communications Review*, 27(3):67–82, July 1997.
[21] M. Mellia, I. Stocia, and H. Zhang. TCP Model for Short Lived Flows. *IEEE Communications Letters*, 6(2), 2002.
[22] J. Nagel. *Congestion Control in IP/TCP Internetworks*, Jan. 1984. RFC 896.
[23] J. Padhye, V.Firoiu, D.Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM SIGCOMM*, 1998.
[24] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low. Congestion Control for High Performance, Stability, and Fairness in General Networks. *IEEE/ACM Trans. Netw.*, 13(1):43–56, 2005.
[25] A. Rangarajan and A. Acharya. ERUF: Early Regulation of Unresponsive Best-Effort Traffic. In *Proceedings of ICNP*, 1999.
[26] J. Roberts. A Survey on Statistical Bandwidth Sharing. *Computer Networks*, 45:319–332, 2004.
[27] S.-C. Yang and G. de Veciana. Bandwidth Sharing: The Role of User Impatience. In *Proceedings IEEE GLOBECOM*, pages 2258–2262, Nov. 2001.
[28] Y. Zhang, S. Kang, and D. Loguinov. Delayed Stability and Performance of Distributed Congestion Control . In *Proceedings of ACM SIGCOMM*, 2004.
[29] Z. Zhao, S. Darbha, and A. L. N. Reddy. A Method for Estimating the Proportion of Nonresponsive Traffic at a Router. *IEEE/ACM Transactions on Networking*, 12(4), Aug. 2004.

APPENDIX: MULTI-USER HOST DETECTION

In this section, we describe a heuristic to distinguish between single-user and multi-user hosts (such as NATs, proxies, remote login servers etc). A host is identified by a unique IP address in the packet trace. In a multi-user host, the sessions generated by different users can be grouped together, making it impossible to distinguish sessions performed by different users. The key observation, however, is that a multi-user host will appear in the trace as generating a very large number of transfers per session, relative to single-user hosts, due to grouping together transfers from different users. This large difference is the key criterion to detect multi-user hosts.

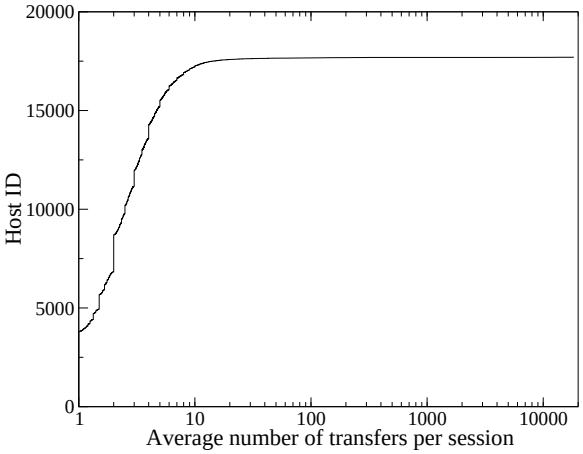

Fig. 10. The average number of transfers per session for each host in the XYZ-in packet trace.

To illustrate this observation, Figure 10 shows the average number of transfers per session for each host (IP address) in the XYZ-in inbound packet trace. Note the logarithmic scale in the x-axis. The vast majority of the hosts generate sessions with up to 10-20 transfers per session. There are only about 100 hosts with a larger number of transfers per session, and some of those hosts generate up to a few thousands of transfers per session. Most likely, those are multi-user hosts. We examined the DNS names of those hosts, and several of them actually indicate proxies and firewalls. So, we chose a threshold of 10 transfers per session, on the average, to distinguish single-user from multi-user hosts. To examine the robustness of the final CTR estimate to the previous threshold, we have calculated the CTR of the same trace as a function of the threshold. It turns out that as long as the threshold is more than 5-6 transfers per session, there is no significant difference in the resulting CTR estimate. This observation increases our confidence in the CTR estimates even if the multi-user host detection heuristic is not very accurate.